

**MODUL 7**  
**LOOPS, VARIABLES, AND ARRAYS**



**Adam Arthur Faizal**

**M3119001**

**TI A**

**PROGRAM STUDI TEKNIK INFORMATIKA**  
**SEKOLAH VOKASI**  
**UNIVERSITAS SEBELAS MARET**  
**SURAKARTA**

**2020**

## **MODUL 7. Loops, Variabels, and Arrays**

### **Capaian Pembelajaran Praktikum:**

1. Menerapkan konsep abstraksi dalam program
2. Menerapkan array dalam program
3. Menerapkan while loop dalam program
4. Menerapkan variabel local dan global

### **Tools:**

Java Development Kit (JDK)  
Greenfoot IDE

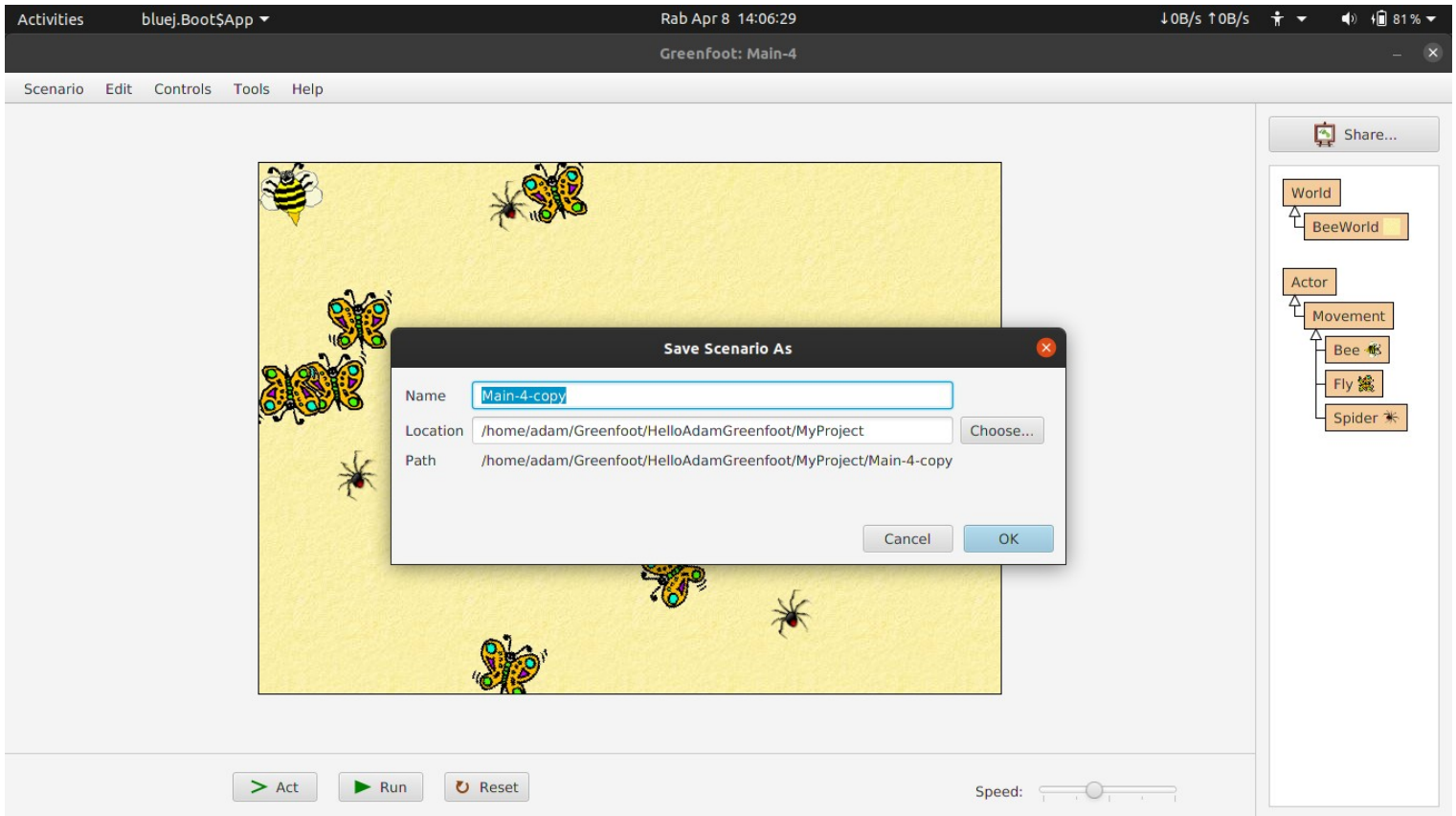
### **Terminologi:**

Isikan terminology yang sesuai untuk definisi dibawah ini:

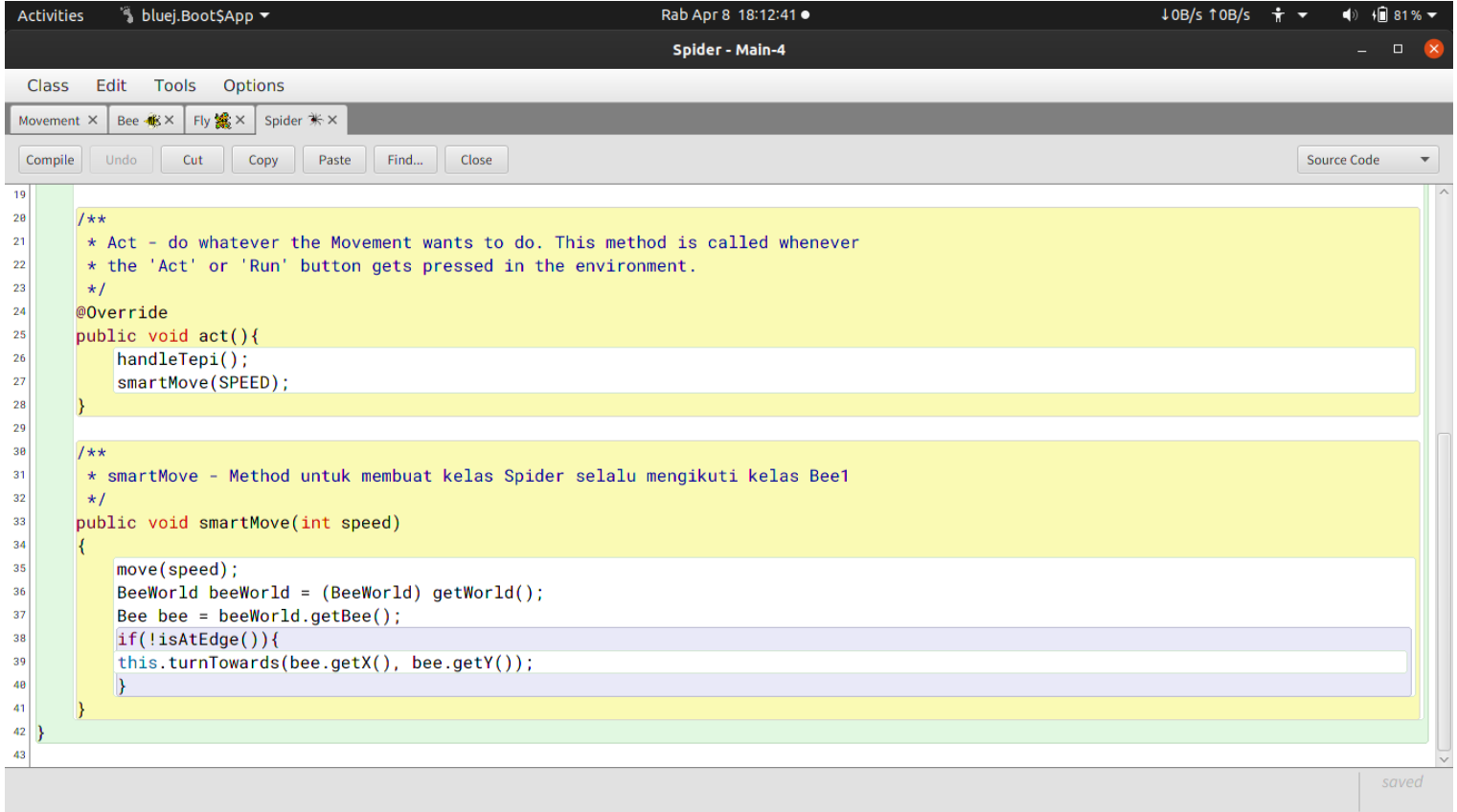
[Loops]	A statement that can execute a section of code multiple times.
[Arrays]	An object that holds multiple variables. An index can be used to access the variables
[Local variables]	A variable declared inside the body of the method to temporarily store values, such as references to objects or integers.

## TRY IT / SOLVE IT:

1. Buka scenario modul sebelumnya, kemudian lakukan versioning dengan melakukan save as. Beri nama folder yang baru untuk menyimpan versi scenario yang baru.



2. Modifikasi program anda, buat method *smartMove()* dimana Spider akan bergerak mengikuti objek Bee. Tambahkan variabel *bee* dan method *getBee()* di class *BeeWorld* sehingga method *smartMove()* pada class *Spider* bisa diimplementasikan. Panggil *smartMove()* di method *act()* kelas *Spider*.

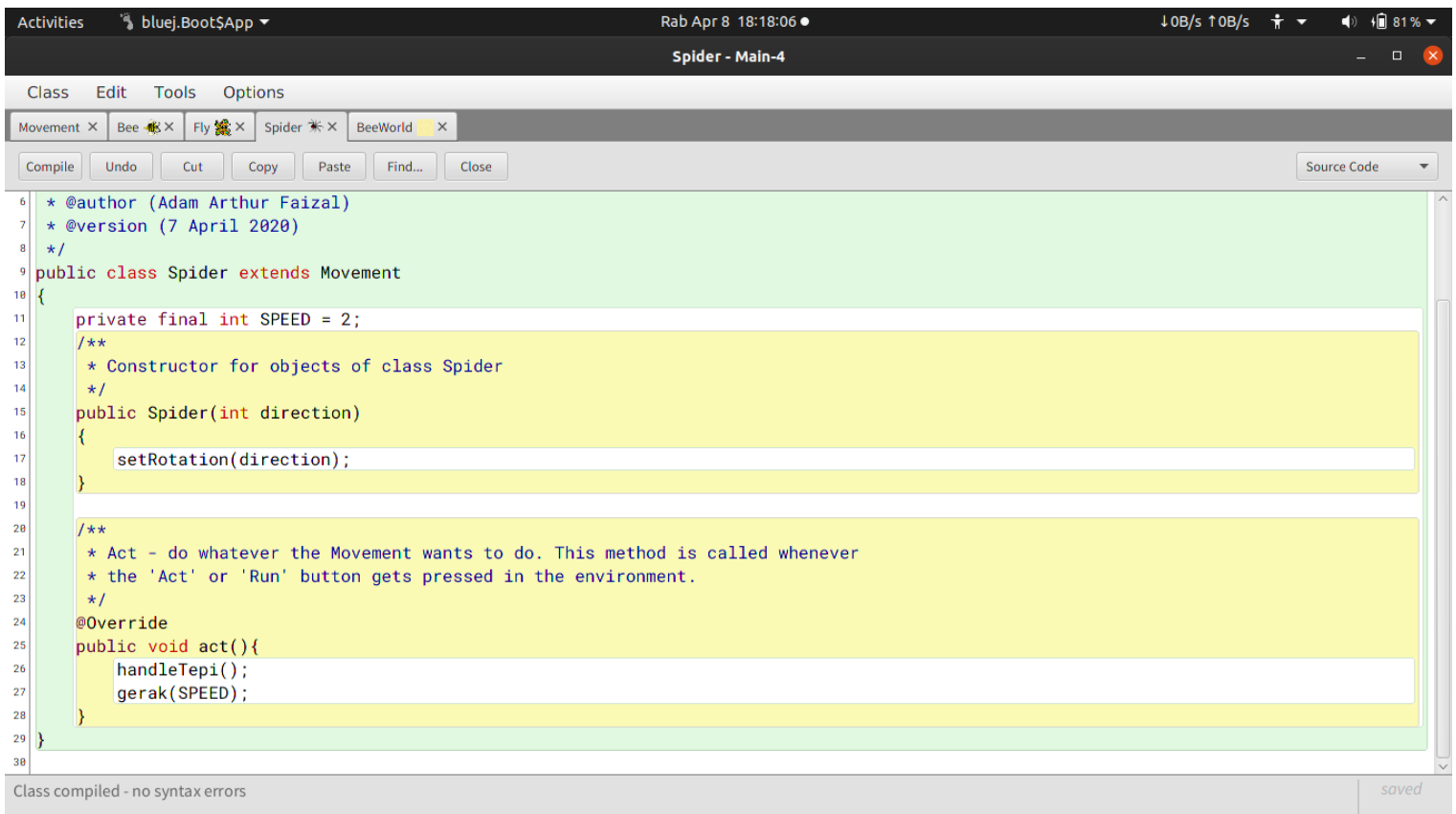


The screenshot shows a Java IDE window titled "Spider - Main-4". The code editor displays the following Java code for the *Spider* class:

```
19
20
21 /**
22  * Act - do whatever the Movement wants to do. This method is called whenever
23  * the 'Act' or 'Run' button gets pressed in the environment.
24  */
25 @Override
26 public void act(){
27     handleTepi();
28     smartMove(SPEED);
29 }
30
31 /**
32  * smartMove - Method untuk membuat kelas Spider selalu mengikuti kelas Bee1
33  */
34 public void smartMove(int speed)
35 {
36     move(speed);
37     BeeWorld beeWorld = (BeeWorld) getWorld();
38     Bee bee = beeWorld.getBee();
39     if(!isAtEdge()){
40         this.turnTowards(bee.getX(), bee.getY());
41     }
42 }
43 }
```

The code is color-coded: comments are in yellow, keywords in red, and identifiers in blue. The IDE interface includes a menu bar (Class, Edit, Tools, Options), a toolbar (Compile, Undo, Cut, Copy, Paste, Find..., Close), and a tab bar showing open files: Movement, Bee, Fly, and Spider. The status bar at the bottom right indicates "saved".

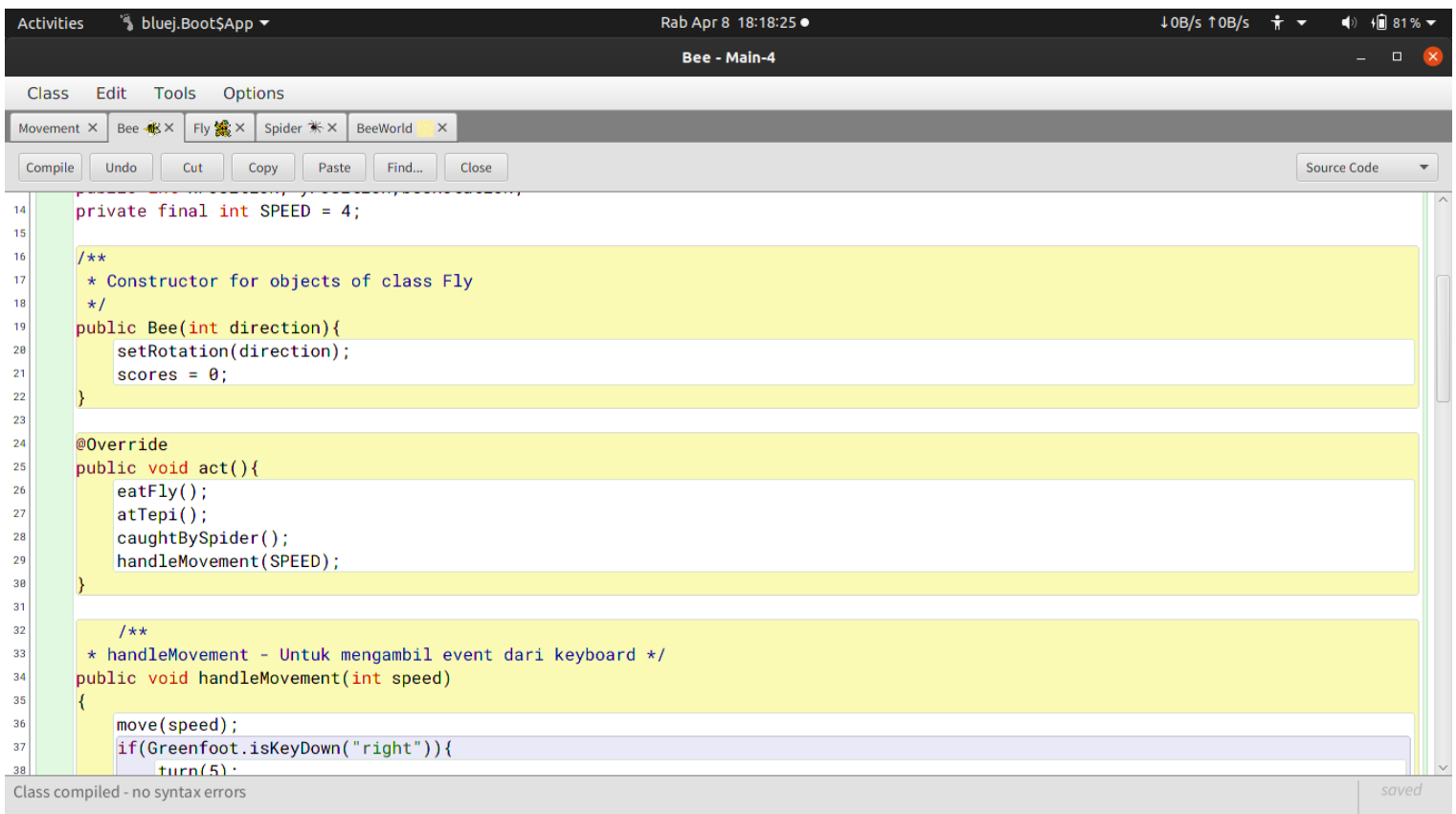
3. Atur ulang variabel speed pada class Spider menjadi 2. Dan set move pada method *handleMovement()* di kelas Bee menjadi 4. Kurangi jumlah objek Spider dari 3 menjadi 1 saja.



The screenshot shows the BlueJ IDE with the 'Spider - Main-4' window active. The code editor displays the source code for the `Spider` class, which extends `Movement`. The code includes a constructor `Spider(int direction)` and an `act()` method. The `SPEED` variable is set to 2. The `act()` method calls `handleTepi()` and `gerak(SPEED)`. The status bar at the bottom indicates 'Class compiled - no syntax errors' and 'saved'.

```
6  * @author (Adam Arthur Faizal)
7  * @version (7 April 2020)
8  */
9  public class Spider extends Movement
10 {
11     private final int SPEED = 2;
12     /**
13      * Constructor for objects of class Spider
14      */
15     public Spider(int direction)
16     {
17         setRotation(direction);
18     }
19
20     /**
21      * Act - do whatever the Movement wants to do. This method is called whenever
22      * the 'Act' or 'Run' button gets pressed in the environment.
23      */
24     @Override
25     public void act(){
26         handleTepi();
27         gerak(SPEED);
28     }
29 }
30
```

Class compiled - no syntax errors saved



The screenshot shows the BlueJ IDE with the 'Bee - Main-4' window active. The code editor displays the source code for the `Bee` class, which extends `Spider`. The code includes a constructor `Bee(int direction)` and an `act()` method. The `SPEED` variable is set to 4. The `act()` method calls `eatFly()`, `atTepi()`, `caughtBySpider()`, and `handleMovement(SPEED)`. The `handleMovement` method is also shown, which calls `move(speed)` and `turn(5)` if the 'right' key is pressed. The status bar at the bottom indicates 'Class compiled - no syntax errors' and 'saved'.

```
14 private final int SPEED = 4;
15
16 /**
17  * Constructor for objects of class Fly
18  */
19 public Bee(int direction){
20     setRotation(direction);
21     scores = 0;
22 }
23
24 @Override
25 public void act(){
26     eatFly();
27     atTepi();
28     caughtBySpider();
29     handleMovement(SPEED);
30 }
31
32 /**
33  * handleMovement - Untuk mengambil event dari keyboard */
34 public void handleMovement(int speed)
35 {
36     move(speed);
37     if(Greenfoot.isKeyDown("right")){
38         turn(5);
39     }
40 }
41
```

Class compiled - no syntax errors saved

4. Modifikasi program anda, tambahkan method *animateBee()* di kelas Bee untuk membuat gerakan objek Bee lebih natural. Terdapat 2 image bee (bee0.png, bee1.png) yang akan digunakan dalam *animateBee()*, dimana Bee akan bergerak dengan bergantian image seakan akan tampak terbang. Gunakan array untuk menyimpan 2 image tersebut. Jangan lupa copy kan image kedalam folder images di scenario anda.



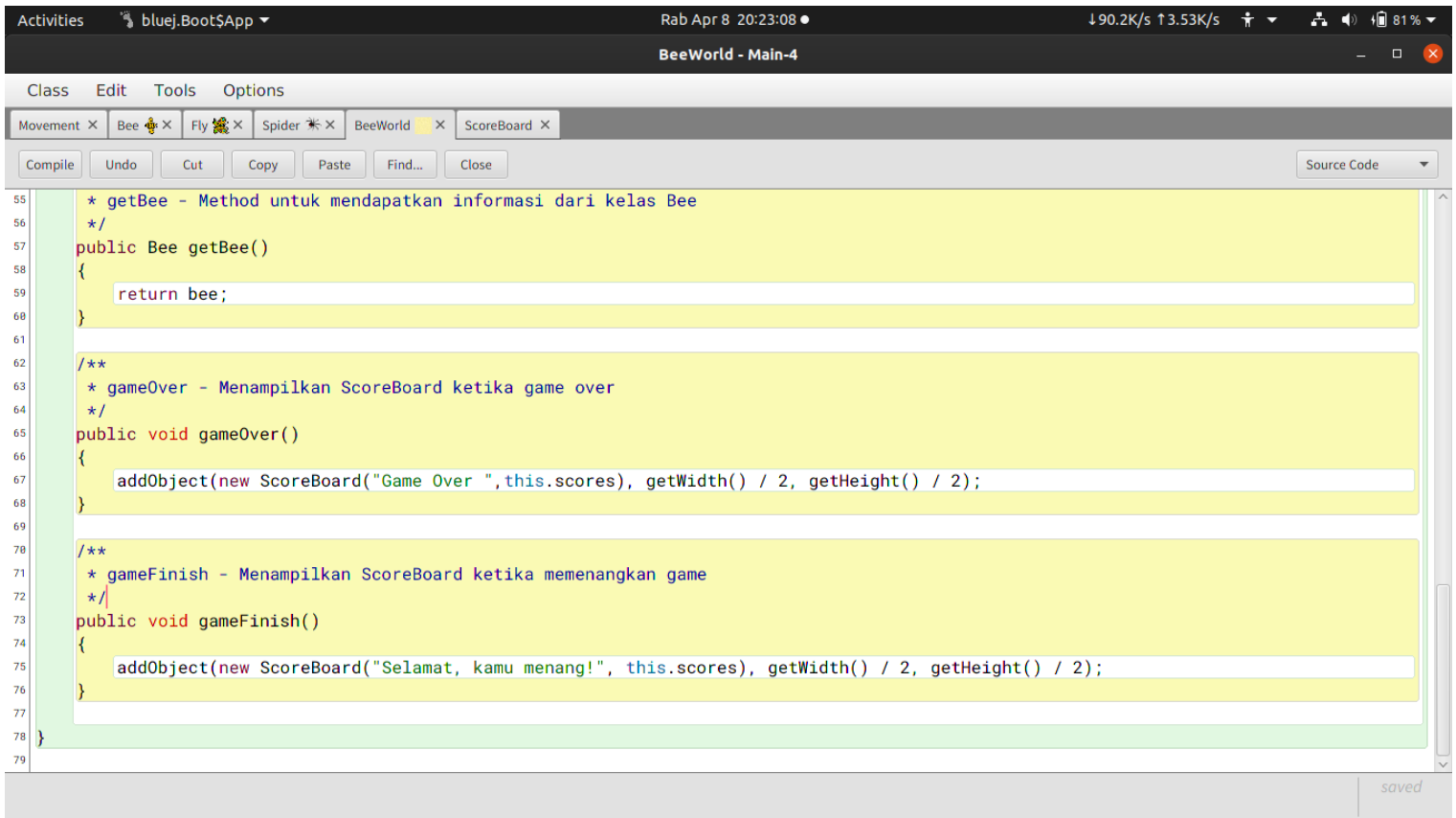
bee0.png



bee1.png

```
Activities bluej.Boot$App ▾ Rab Apr 8 19:43:05 • ↓0B/s ↑0B/s 81% ▾
Bee - Main-4
Class Edit Tools Options
Movement × Bee × Fly × Spider × BeeWorld ×
Compile Undo Cut Copy Paste Find... Close Source Code ▾
189 /**
110  * loopingArrays - Melakukan looping pada array images
111  */
112 public void loopingArrays()
113 {
114     for (int i = 0; i < this.images.length; i++){
115         this.images[i] = new GreenfootImage("bee" + i + ".png");
116     }
117 }
118
119 /**
120  * animateBee - Membuat kelas Bee seakan-akan bergerak secara dinamis
121  */
122 public void animateBee()
123 {
124     if (this.currentImages == this.images.length){
125         this.currentImages = 0;
126         return;
127     }
128     setImage(this.images[this.currentImages]);
129     this.currentImages++;
130 }
131
132 }
133
Class compiled - no syntax errors saved
```

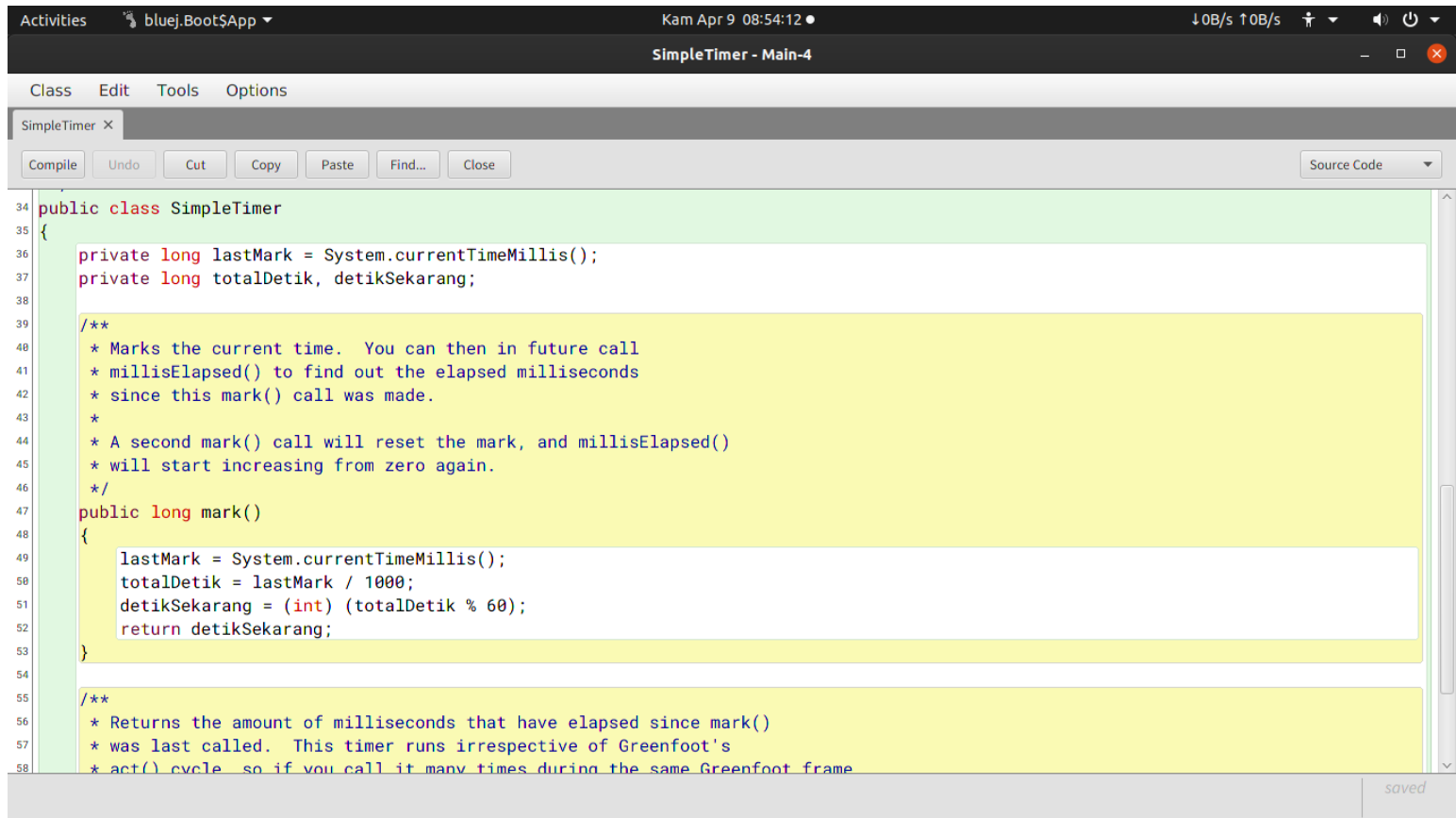
5. Tambahkan source code untuk mengakhiri permainan/scenario jika nilai score mencapai 50 dan munculkan kalimat “ SELAMAT ANDA MENANG” di bagian tengah world.



```
55  * getBee - Method untuk mendapatkan informasi dari kelas Bee
56  */
57  public Bee getBee()
58  {
59      return bee;
60  }
61
62  /**
63   * gameOver - Menampilkan ScoreBoard ketika game over
64   */
65  public void gameOver()
66  {
67      addObject(new ScoreBoard("Game Over ",this.scores), getWidth() / 2, getHeight() / 2);
68  }
69
70  /**
71   * gameFinish - Menampilkan ScoreBoard ketika memenangkan game
72   */
73  public void gameFinish()
74  {
75      addObject(new ScoreBoard("Selamat, kamu menang!", this.scores), getWidth() / 2, getHeight() / 2);
76  }
77  }
78
79  }
```


Tantangan:

6. Tambahkan fitur timer untuk game/scenario. Jika timer habis maka permainan berakhir.



The screenshot shows the SimpleTimer.java source code in an IDE. The code defines a SimpleTimer class with a mark() method and a comment for millisElapsed().

```
34 public class SimpleTimer
35 {
36     private long lastMark = System.currentTimeMillis();
37     private long totalDetik, detikSekarang;
38
39     /**
40      * Marks the current time. You can then in future call
41      * millisElapsed() to find out the elapsed milliseconds
42      * since this mark() call was made.
43      *
44      * A second mark() call will reset the mark, and millisElapsed()
45      * will start increasing from zero again.
46      */
47     public long mark()
48     {
49         lastMark = System.currentTimeMillis();
50         totalDetik = lastMark / 1000;
51         detikSekarang = (int) (totalDetik % 60);
52         return detikSekarang;
53     }
54
55     /**
56      * Returns the amount of milliseconds that have elapsed since mark()
57      * was last called. This timer runs irrespective of Greenfoot's
58      * act() cycle so if you call it many times during the same Greenfoot frame
```



The screenshot shows the BeeWorld.java source code in an IDE. The code includes a showTimer() method and an updateScore() method.

```
43 }
44
45 /**
46  * showTimer - Menampilkan timer di layar
47  */
48 public void showTimer()
49 {
50     this.detik = timer.mark();
51     showText("Timer : " + this.detik, 350, 480);
52     if (this.detik == 60){
53         Greenfoot.stop();
54     }
55 }
56
57 /**
58  * updateScore - Method untuk memperbarui nilai score ketika kelas Bee memakan kelas Fly
59  */
60 public void updateScore()
61 {
62     this.scores++;
63     showText("Score : " + scores, 60, 480);
64     if (this.scores == 50){
65         gameFinish();
66     }
67 }
```



```
122
123
124 /**
125  * animateBee - Membuat kelas Bee seakan-akan bergerak secara dinamis
126  */
127 public void animateBee()
128 {
129     if (this.currentImages == this.images.length){
130         this.currentImages = 0;
131         return;
132     }
133     setImage(this.images[this.currentImages]);
134     this.currentImages++;
135 }
136
137 /**
138  * showTimer - Menampilkan timer di layar
139  */
140 public void showTimer()
141 {
142     BeeWorld beeWorld = (BeeWorld) getWorld();
143     beeWorld.showTimer();
144 }
145 }
146
```

Class compiled - no syntax errors saved

***Setelah sesi praktikum SELESAI, laporan praktikum dan source code (zip) harus dikirim/diupload ke google classroom sebelum pertemuan berikutnya.***